US Army Corps
of Engineers
Construction Engineering
Research Laboratory

USA-CERL TECHNICAL REPORT M-89/03
January 1989
Advanced Technology Applications for Quality Assurance

2

# Quality Assurance/Quality Control in Construction: Expert Systems Development for Multilevel Experienced Users

by
Thomas Gatton
Debbie Lawrence
Frank Kearney

Within the U.S. Army Corps of Engineers (USACE) military construction program, a limited number of quality assurance/quality control (QA/QC) personnel are responsible for an increasingly large workload involving many more complex practices than found in traditional construction. To ensure the continued quality of military facilities, several approaches are being considered, including automation. Expert systems technology in particular shows great promise in creating tools to assist QA/QC elements.

Past development of expert systems has shown that the user must become involved early in the process to ensure suitable system performance. However, in developing some systems, the user is not known and the system cannot be tailored for a particular level of domain knowledge. When this situation occurs, it is necessary to provide flexibility in a system to handle users with differing levels of knowledge about the domain. Incorporating this flexibility into an expert system is a major problem in current expert system development and different approaches have been tried to deal with the problem.

This report describes the development of decision models to allow flexibility in handling multilevel user knowledge about a specific domain. These models can be implemented in many different commercially available shells to supplement the available options and provide a more flexible interface for expert system users.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

## REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188
Exp Date Jun 30 1986

| | |
|---|---|
| 1a REPORT SECURITY CLASSIFICATION<br>Unclassified | 1b RESTRICTIVE MARKINGS |
| 2a SECURITY CLASSIFICATION AUTHORITY | 3 DISTRIBUTION / AVAILABILITY OF REPORT<br>Approved for public release; distribution is unlimited. |
| 2b DECLASSIFICATION / DOWNGRADING SCHEDULE | |
| 4 PERFORMING ORGANIZATION REPORT NUMBER(S)<br>USA-CERL TR M-89/03 | 5 MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a NAME OF PERFORMING ORGANIZATION<br>U.S. Army Construction Engr Research Laboratory<br>6b OFFICE SYMBOL (If applicable)<br>CECER-EM | 7a NAME OF MONITORING ORGANIZATION |
| 6c ADDRESS (City, State, and ZIP Code)<br>P.O. Box 4005<br>Champaign, IL 61820-1305 | 7b ADDRESS (City, State, and ZIP Code) |
| 8a NAME OF FUNDING / SPONSORING ORGANIZATION<br>HQUSACE<br>8b OFFICE SYMBOL (If applicable)<br>CEEC-CE | 9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c ADDRESS (City, State, and ZIP Code)<br>20 Massachusetts Ave, N.W.<br>Washington, D.C. 20314-1000 | 10 SOURCE OF FUNDING NUMBERS |

| PROGRAM ELEMENT NO | PROJECT NO | TASK NO | WORK UNIT ACCESSION NO |
|---|---|---|---|
| 4A162731 | AT41 | BO | 042 |

11 TITLE (Include Security Classification)
Quality Assurance/Quality Control in Construction: Expert Systems Development for Multilevel Experienced Users (U)

12 PERSONAL AUTHOR(S)
Gatton, Thomas; Lawrence, Debbie; Kearney, Frank

| 13a TYPE OF REPORT | 13b TIME COVERED | 14 DATE OF REPORT (Year, Month, Day) | 15 PAGE COUNT |
|---|---|---|---|
| final | FROM ______ TO ______ | 1989, January | 28 |

16 SUPPLEMENTARY NOTATION
Copies are available from the National Technical Information Service
Springfield, VA 22161

17 COSATI CODES

| FIELD | GROUP | SUB-GROUP |
|---|---|---|
| 12 | 09 | |
| 13 | 13 | |

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
quality assurance
expert system
quality control
construction

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Within the U.S. Army Corps of Engineers (USACE) military construction program, a limited number of quality assurance/quality control (QA/QC) personnel are responsible for an increasingly large workload involving many more complex practices than found in traditional construction. To ensure the continued quality of military facilities, several approaches are being considered, including automation. Expert systems technology in particular shows great promise in creating tools to assist QA/QC elements.

Past development of expert systems has shown that the user must become involved early in the process to ensure suitable system performance. However, in developing some systems, the user is not known and the system cannot be tailored for a particular level of domain knowledge. When this situation occurs, it is necessary to provide flexibility in a system to handle users with differing levels of knowledge about the domain. (Cont'd)

| | |
|---|---|
| 20 DISTRIBUTION / AVAILABILITY OF ABSTRACT<br>☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT] ☐ DTIC USERS | 21 ABSTRACT SECURITY CLASSIFICATION<br>Unclassified |
| 22a NAME OF RESPONSIBLE INDIVIDUAL<br>Dana Finney | 22b TELEPHONE (Include Area Code)<br>(217) 352-6511 ext 389<br>22c OFFICE SYMBOL<br>CECER-IMT |

DD FORM 1473, 84 MAR  83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Block 19. (Cont'd.)

Incorporating this flexibility into an expert system is a major problem in current expert system development and different approaches have been tried to deal with the problem.

This report describes the development of decision models to allow flexibility in handling multilevel user knowledge about a specific domain. These models can be implemented in many different commercially available shells to supplement the available options and provide a more flexible interface for expert system users. (SDW)

## FOREWORD

This investigation was performed for the Directorate of Engineering and Construction, Headquarters, U.S. Army Corps of Engineers (HQUSACE), under Project 4A162731AT41, "Military Facilities Engineering Technology"; Work Unit BO-042, "Advanced Technology Applications for Quality Assurance." The HQUSACE Technical Monitors were Robert Chesi and Richard Carr, CEEC-CE.

This work was performed by the Engineering and Materials Division (EM) of the U.S. Army Construction Engineering Research Laboratory (USA-CERL). Dr. Robert Quattrone is Chief, EM. The USA-CERL technical editor was Dana Finney, Information Management Office.

COL Carl O. Magnell is Commander and Director of USA-CERL, and Dr. L. R. Shaffer is Technical Director.

## CONTENTS

# QUALITY ASSURANCE/QUALITY CONTROL IN CONSTRUCTION: EXPERT SYSTEMS DEVELOPMENT FOR MULTILEVEL EXPERIENCED USERS

# 1 INTRODUCTION

## Background

The state of the economy has forced the construction industry to optimize its operations where possible. The U.S. Army Corps of Engineers (USACE) military construction program, too, is facing this challenge as defense spending cutbacks are likely to continue. Compounding the economic situation is that advances in technology have increased construction rates as well as created more complex construction practices in some cases.

The impact of these conditions on construction quality has been accentuated by a declining number of quality assurance/quality control (QA/QC) staff. To supplement the capabilities and improve the productivity of QA/QC personnel, many emerging computer technologies are being investigated. One very promising approach is the use of expert systems, a new computer technology that allows an "expert's" knowledge to be encoded for later use by persons with less expertise. An expert system allows a user to access the expert's knowledge rapidly and easily.

As an example, an expert system could be developed for roof construction. This system would contain roofing knowledge from both the literature and experts' minds. The system could be used by engineers or others to supplement their experience in one of three modes: (1) as a tutor, by asking the user questions and explaining why the questions are asked, what conditions are necessary for certain problems to occur, etc., (2) as a backup to reinforce a "gut feeling," (e.g., that a certain condition was the problem), and (3) as an analyst of all factors affecting the situation, i.e., many factors may combine to cause a problem and it is difficult to analyze all of their effects and impacts without having an expert's knowledge on the topic.

The development of expert systems for field applications must take into account the user's level of expertise in the domain. Often, however, all potential users' knowledge about the domain ranges from a limited amount to a level approaching that of an expert. Thus, during the development of many expert systems, the precise level of user knowledge is unknown and may vary widely for a particular group of users.

This wide range of users' familiarity with the domain introduces problems with systems that offer only heuristic knowledge (see Chapter 2). While systems that provide expert heuristics may be suitable to users who understand the domain, those who are not as knowledgeable may have difficulty in understanding many of the explanations and lines of reasoning that the expert system may provide. There is a great demand for a method of introducing flexibility into an expert system so that different levels of users can be accommodated.

## Objective

The objective of this study is to assess the current state of the art in development of expert systems for a user group with a wide spectrum of domain knowledge and to develop models for designing such systems.

## Approach

The literature was surveyed for current research dealing with multilevel domain knowledge across a user group. Next, decision models were developed to handle the different levels of knowledge and then implemented in a prototype expert system. Finally, the models and implementation were critiqued in view of available expert systems technology. Future needs were also assessed to identify research directions that offer potential in solving some of the problems in application.

## Scope

This work is aimed at identifying methods of developing expert systems for user groups with a wide range of domain knowledge, including those for which the user groups are not known. This study is not intended to be an introduction to expert systems, nor does it attempt to identify a general solution applicable to all existing expert system shells.

## Mode of Technology Transfer

Information in this report will be used in developing expert systems-based knowledge acquisition and knowledge engineering tools for QA/QC in military construction. When these tools have been completed and pilot-tested successfully, they will be transferred to the field through onsite training, tutorial, and hands-on experience. In addition, they may be transferred to industry through mechanisms such as Cooperative Research and Development Agreements.

## 2 LEVEL OF USER DOMAIN KNOWLEDGE

One of the leading researchers in the area of expert systems, Edward Feigenbaum, has given the following definition of an expert system:[1]

> [An expert system is] an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. Knowledge necessary to perform at such a level, plus the inference procedures used, can be thought of as a model of the expertise of the best practitioners of the field.
>
> The knowledge of an expert system consists of facts and heuristics. The 'facts' constitute a body of information that is widely shared, publicly available, and generally agreed upon by experts in a field. The 'heuristics' are mostly private, little-discussed rules of good judgment (rules of plausible reasoning, rules of good guessing) that characterize expert-level decision making in the field. The performance level of an expert system is primarily a function of the size and the quality of a knowledge base it possesses.

The first step in developing an expert system is to identify (1) the limited topic area, or domain, for which the system is needed and (2) identify tne users. If the domain is too large, the resulting system might never be completed and not include all of the rules necessary to cover that domain, rendering the system less than adequate for its purpose. It is helpful to know who the users will be so that explanations contained within the system and the user interfaces are adequate. The user must feel comfortable with the system through its interface as well as with the presentation of the intended information. Once these two items are identified, development of a useful system can begin.

Expert systems are composed of two main parts: (1) the knowledge base and (2) the inference engine. The knowledge base contains information in the form of facts and heuristics about the particular domain. This information is gathered by a process called "knowledge acquisition" and is converted into "rules" that can be understood by the computer. The second part, in the inference engine, looks at the knowledge base and decides which questions to ask the system users to help them solve their particular problem. In operation, system users are prompted by the computer for information about the situation. Based on the answers, the expert system will be able to reach a conclusion and explain why that conclusion was reached. The user may also query the system during consultation about why it is asking particular questions.

The quality of an expert system is reflected in both its ability to make proper decisions and to explain those decisions in a way that is understandable by the user.[2] However, many of the explanation facilities of existing systems leave considerable room for improvement in these areas.[3] One reason for the perception that explanation levels are poor may be the user's expectations and the level of information required to explain an expert's reasoning.

[1]A. Barr and E. A. Feigenbaum, *The Handbook of Artificial Intelligence*, Vols I through III (Heuristech Press, Stanford, CA, 1981).

[2]E. H. Shortcliffe, *Computer-Based Medical Consultations: MYCIN* (North-Holland, Amsterdam, 1976).

[3]D. C. Berry and D. E. Broadbent, "Expert Systems and the Man-Machine Interface," *Expert Systems* (October 1986).

Knowledge levels in expert systems are classified as two types: (1) surface or shallow knowledge and (2) deep knowledge.[4] Surface knowledge is of the type gained by an expert over a period of time through experience with a particular domain. It consists of heuristics or "rules of thumb" that the expert uses in making decisions. Deep knowledge, on the other hand, is more like the formal knowledge gained from textbooks and other educational experiences. Often, when experts elucidate their knowledge, the fundamental scientific principles on which a decision is based have been forgotten and the resulting reasoning that becomes encoded into the expert system is shallow and does little to help some users understand how a particular decision was reached. The lack of deep knowledge is one of the major complaints from persons using today's expert systems.[5] The adequacy of explanations for a particular user is directly related to the domain knowledge which that user has.

Past development of expert systems has shown that the user must become involved early in the process to ensure suitable system performance.[6] However, in developing some systems, the potential user is not known and the system cannot be tailored for a particular user level. In this case, it is necessary to design the system with enough flexibility to handle users who are at different levels of domain knowledge. This is a major issue in creating expert systems today and different approaches have been adopted to deal with it.

One method of accommodating multiple levels of user domain knowledge is to instruct users at the beginning of the session to assign themselves a knowledge level. The system then selects responses for explanations and reasoning appropriate to that level. Another technique is to have the system ask the user questions about the domain at the beginning of a session, and then determine the user's level of expertise from the answers given. There is also the more complex approach of developing a "user model" that learns what the user knows and tailors its responses accordingly.[7] However, this work is in the conceptual stage, with only very simple models developed to date.

All previous expert systems development for real applications have assumed that the users' level of expertise must be taken into account. In contrast, the approach adopted in this study is to consider methods of developing expert systems that would be independent of the user's domain knowledge.

[4]P. Harmon and D. King, *Expert Systems* (John Wiley and Sons, 1985).

[5]B. G. Buchanan, "Expert Systems: Working Systems and the Research Literature," *Expert Systems* (January 1986).

[6]D. C. Berry and D. E. Broadbent.

[7]D. C. Berry and D. E. Broadbent.

# 3 CONSULTATION AND TUTORIAL EXPERT SYSTEMS

Expert systems are distinguished from other types of software by their basic structure and by the way information is presented to the user. As mentioned in Chapter 2, an expert system consists of a knowledge base containing encoded knowledge and an inference engine that uses the knowledge base in reasoning about a particular problem. Expert systems are unique in that, first, the knowledge is separate from the inference mechanism and, second, they are able to explain their reasoning. Several packages referred to as "shell" systems are available that provide the inference engine but allow the programmer to "plug in" a knowledge base for a particular application. These systems reduce the time required for system development but also limit the programmer's flexibility and options in designing the system. Shell systems are largely based on production rules, which are of the form:

IF <condition> THEN <action>,

where the condition can be further expanded to include logical conditions and implications. This is the primary rule of inference from which production rules are based and is called *modus ponens.*[8]

In addition to the inference engine, most shells have a compiler that allows easy modification and addition to the knowledge base. The syntax for each shell is unique and may offer specialized features that allow for optional types of control. In this way, the programmer has some flexibility in tailoring the system for the user. All shells have the basic ability to represent rules, and second-generation shells have increasingly improved their explanation facilities over earlier versions.

The areas of application for expert systems and the types of problems involved are:[9]

- Interpretation--inferring situation descriptions from sensor data
- Prediction--inferring likely consequences of given situations
- Design--configuring objects under constraints
- Planning--designing actions
- Monitoring--comparing observations to plan vulnerabilities
- Debugging--prescribing remedies for malfunctions
- Repair--executing a plan to administer a prescribed remedy
- Instruction--diagnosing, debugging, and correcting user behavior
- Control--interpreting, predicting, repairing, and monitoring system behaviors.

[8]B. G. Buchanan and R. O. Duda, *Principles of Rule-Based Expert Systems*, Fairchild Technical Report No. 626.
[9]F. Hayes-Roth and D. A. Waterman, *Building Expert Systems* (Addison-Wesley, 1983).

This list points to the lack of integrated instructional knowledge in expert systems and separates instructional applications as unique. However, there are expert systems that provide not only expert-level knowledge but also the deep knowledge that reasons and explains using basic principles. These types of systems have been categorized as "intelligent computer-aided instruction (ICAI)," although some of them have been implemented as expert systems.

An ICAI system's main components are the problem-solving expertise that is to be transferred to the user, the model of the user's knowledge, and strategies for tutoring the user.[10] One example of an expert system used for this purpose is GUIDON which was developed by William Clancey at Stanford University.[11] Conceptually, this system is the reversal of MYCIN, an expert system for medical diagnosis of blood diseases, in that it gives the user a patient to diagnose and then acts as tutor during the process. GUIDON has all of the knowledge that MYCIN contains and also uses all case experiences on which MYCIN had consulted. To reverse the system, it was necessary to build an inferencing mechanism that would deal with the program's tutoring aspects. When the user would ask a question about the patient, the system would analyze the relevance of the question and react accordingly. If the user made a diagnosis without enough information, the system would properly tutor the user to see the mistake. Again, though, as is the problem with other expert systems, the user was assumed to be at a certain level and basic definitions were not provided. However, GUIDON is an example of a system in which one of the two essential requirements was satisfied--that of providing the proper level of process tutoring.

Another system, developed by Allan Collins and Albert Stevens and named WHY, deals with both processes and factual information.[12] The system tutors users in the causes of rainfall, which is a complex process involving a high level of factual information. In this system, production rules are made to represent tutoring heuristics. The user's misconceptions are categorized into a set of goals that will provide questions and answers to assist him/her in clarifying areas of weakness.

One of the main distinctions between these systems and commercial expert systems is that of the developmental environment. These systems were created using the resources of academia and the programs were developed from the ground up, so to speak. This process differs from shell system development in that the programmer who uses a shell system constructs only the knowledge base and has no access to the inferencing mechanism. Although the shell allows quicker development of a prototype system, there is less flexibility to adapt the features to a particular application. The type of user interface that occurs is subject to the shell's limitations as well as the programmer's style. A method of using commercially available shell systems to improve the user interface and, therefore, the explanation facilities is needed. In particular, when the users have differing levels of domain knowledge, a method of providing system flexibility is needed. Such an improvement would allow the programmer to enter both process knowledge and factual knowledge that would be available to the user upon request and in a form consistent with the user's knowledge about the domain.

[10]A. Barr and E. A. Feigenbaum.
[11]P. Harmon and D. King.
[12]A. Barr and E. A. Feigenbaum.

# 4 A MODEL FOR HANDLING USERS' MULTILEVEL DOMAIN KNOWLEDGE

## Misconceptions and Their Remedies

In a tutoring expert system, there are five types of misconceptions a student can make, each with its own remedial activity:[13]

1. Factual bugs. The tutor deals with these misconceptions by correcting the student. The teaching of facts is not the goal of tutoring; interrelations of facts are more important.

2. Outside-domain bugs. These misconceptions are about causal structure, which the tutor chooses not to explain in detail.

3. Overgeneralization. When a student makes a general rule from an insufficient set of factors, the tutor will find counterexamples to probe for more factors.

4. Overdifferentiation. When a student counts factors as necessary when they are not, the tutor will generate counterexamples to show that they are not.

5. Reasoning bugs. Tutors will attempt to teach students skills such as forming and testing hypotheses and collecting enough information before drawing a conclusion.

These five misconceptions can be grouped into two basic categories:

- Reasoning errors. The user does not understand the relationships between the parameters and how they are used to reach a conclusion.

- Factual errors. The user does not understand basic definitions and elements associated with the domain.

The types of errors differ for each user, particularly when a wide or unknown range of domain knowledge exists across the user spectrum. Although explanation facilities are provided with commercially available shell systems, they may be insufficient for some users. Similarly, the level of reasoning that an expert provides for a conclusion may not be clear to all users. A solution to these two user interface problems must include greater depth in the knowledge base as well as effective organization of the rules to provide clarity at a level comfortable to the user.

## Example Tutorial System

To address the above issues, an example is helpful in demonstrating the basic concepts involved. Consider the problem of diagnosing the malfunction of an automobile that will not start. The following conclusions are possible, depending on the conditions of various components within the automobile:

- The battery is dead or lacks enough charge

[13]A. Barr and E. A. Feigenbaum.

- The starter is broken
- The car is out of gas
- The fuel system is faulty
- The carburetor is malfunctioning
- The ignition system is faulty
- There are other mechanical problems.

The following parameters, definitions, and values are necessary to identify the malfunction:

---> What is the condition of the starter?

(1) The starter is working normally
(2) The starter is turning slowly
(3) The starter is clicking
(4) The starter is not doing anything.

Parameter definition. The starter is an electrical motor that turns over the engine in order to start it. Typically, the ignition key must be turned clockwise as far as possible to activate the starter. If the starter is working normally, the motor will emit a mild, rather uniform grinding noise. If the noise seems to speed up and slow down noticeably, then the engine is probably turning too slowly. If there is only a clicking noise or no noise at all when the starter is engaged, it is not working properly.

---> Are the lights working normally?

(1) Yes, the lights are working normally
(2) No, the lights are dimmer than usual
(3) No, the lights are not working at all.

Parameter definition. The "lights" refer to the headlights of the automobile. When they are working normally, they emit a bright, white light. When there is a malfunction, they will be dim and emit a yellowish light or no light at all.

---> Does the gas gauge indicate that there is any gas?

(1) Yes, it shows that there is gas in the car
(2) No, it shows that there is no gas.

Parameter definition. The gas gauge has two letters on it: E and F. If the needle is pointing to the E, it indicates that there is no gas in the tank (i.e., "empty"). If the needle is pointing between the E and F or is pointing to the F (i.e., "full"), then there is gas in the tank.

---> Is gas reaching the carburetor?

(1) Yes, gas is going into the carburetor
(2) No, gas is not going into the carburetor.

Parameter definition. Gasoline goes into the carburetor through the fuel line. This condition can be checked by disconnecting the fuel line at the carburetor and observing if any gas is present. During a start, gas should pump out of the fuel line, indicating that it is reaching the carburetor.

—> Is any spark present at the sparkplugs?

(1) Yes, there is spark at the sparkplugs
(2) No, there is no spark at the sparkplugs.

Parameter definition. The presence of a spark at the sparkplugs can be detected by a small penlike instrument that is placed next to the sparkplug wire. If spark is present, the instrument will glow. Alternatively, if the wire is removed from the sparkplug and positioned within a quarter inch of the sparkplug tip, the spark will be observable when cranking the engine. An observable spark here also indicates the presence of spark.

Rules that will be used to diagnose the problem are:

```
1. IF    (the starter is not working normally)
   AND   (the lights are not working normally)
   THEN  [Check the battery and its connections].

2. IF    (the starter is not working normally)
   AND   (the lights are working normally)
   THEN  [Check the starter and its voltages].

3. IF    (the starter is working normally)
   AND   (the gas gauge indicates that there is no gas)
   THEN  [Put gas in the tank].

4. IF    (the starter is working normally)
   AND   (the gas gauge indicates that there is gas)
   AND   (no gas is going into the carburetor)
   THEN  [Check the fuel system for malfunction].

5. IF    (the starter is working normally)
   AND   (the gas gauge indicates that there is gas)
   AND   (gas is going into the carburetor)
   AND   (there is no spark present at the spark plugs)
   THEN  [Check the ignition system for a malfunction].

6. IF    (the starter is working normally)
   AND   (the gas gauge indicates that there is gas)
   AND   (gas is going into the carburetor)
   AND   (there is spark present at the spark plugs)
   THEN  [Check the engine for mechanical problems].
```

A decision tree representing these rules is shown in Figure 1.

For this simple demonstration, backward chaining will be used and the rules will be ordered as indicated above. The following conventions will be used:

- ? - Question to obtain parameter value
- ( ) - Selected parameter value

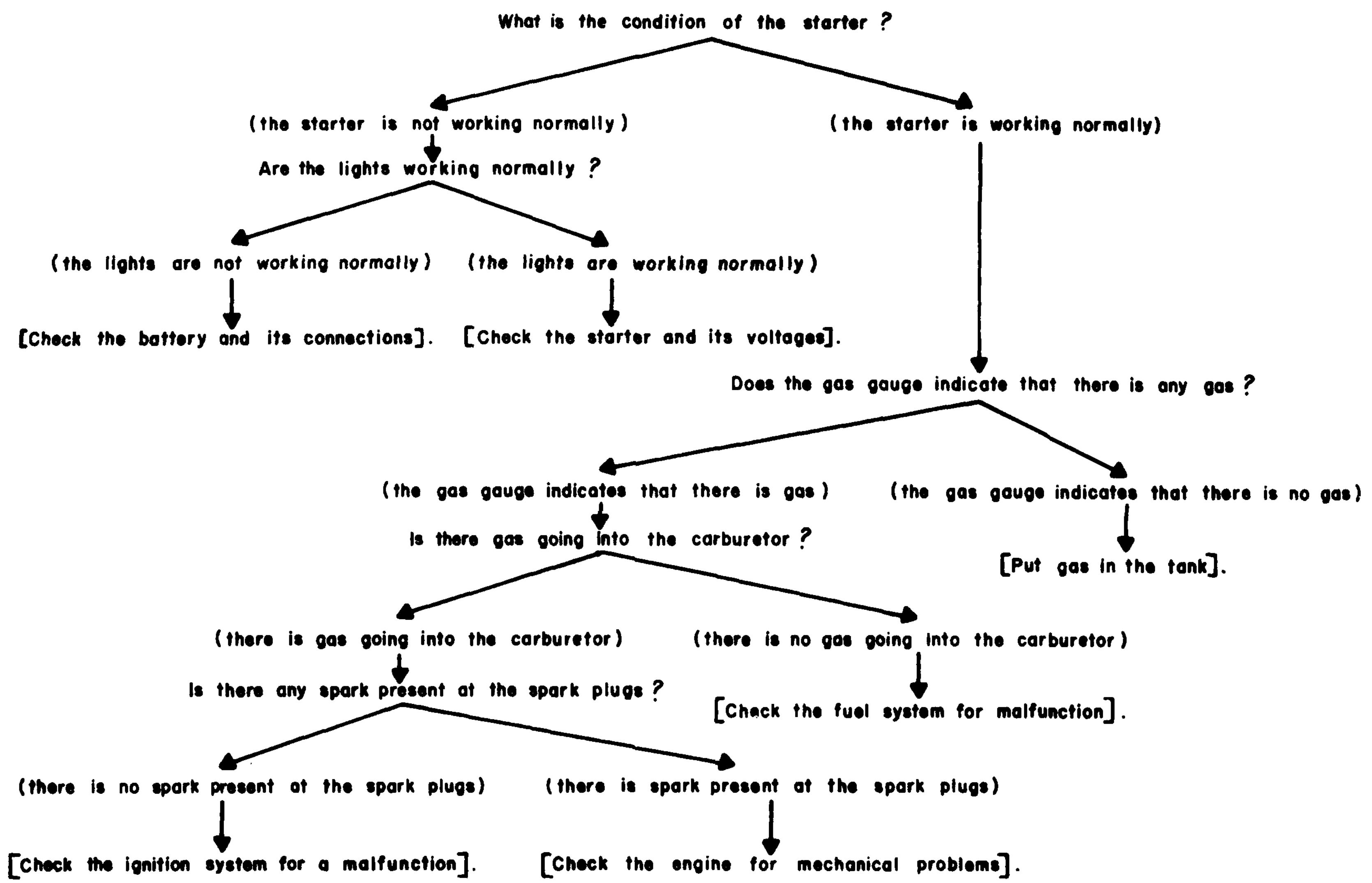
What is the condition of the starter ?
(the starter is not working normally)
Are the lights working normally ?
(the lights are not working normally)
[Check the battery and its connections].
(the lights are working normally)
[Check the starter and its voltages].
(the starter is working normally)
Does the gas gauge indicate that there is any gas ?
(the gas gauge indicates that there is gas)
Is there gas going into the carburetor ?
(there is gas going into the carburetor)
Is there any spark present at the spark plugs ?
(there is no spark present at the spark plugs)
[Check the ignition system for a malfunction].
(there is spark present at the spark plugs)
[Check the engine for mechanical problems].
(there is no gas going into the carburetor)
[Check the fuel system for malfunction].
(the gas gauge indicates that there is no gas)
[Put gas in the tank].

**Figure 1. A simple expert system.**

- [ ] - Conclusion or recommendation.
- _ - Flow of control.

At the beginning, the system will try to prove the first rule by prompting the user for the condition of the starter. The user may ask for an explanation of the question or the reason that the question is necessary. The explanation is the same as that denoted by "parameter definition" above. The reason is derived from the rule that the system is trying to prove. At this point, the system will answer something similar to:

I am trying to determine if the battery is dead, which would be indicated
IF:
The starter is not working normally
AND:
The lights are not working normally.

This explanation is developed directly from the rules and parameter definitions. Likewise, if the user indicates that the starter is not working normally and the lights are not working normally, the system will give the following conclusion:

Conclusion: the battery is dead or undercharged.

At this point, the user may wish to ask the system how or why it reached this conclusion. The system will generate an answer from the rule that led to this conclusion:

When the lights are not working normally and
the starter is not working normally, then
it is usually the case that the battery is dead
or has too little charge.

Many expert systems associate statistical values with their conclusions and allow probabilities to be associated with levels of confidence in the answer that the user gives. Although these options can indicate accuracy, they do not affect the basic explanation facilities used. In this example, the user would have the basic interface shown here and answers would reflect current rules and applicable parameter values.

When the user reaches rule 4, it is questionable whether he or she would be able to perform the check for gasoline going into the carburetor. Any mechanic (professional or backyard) would have little trouble in checking this condition, but if the user is a typical layman, the explanation facilities would do little to help in accomplishing the task. At this point, the user would lack factual data about the engine and should be tutored through information that would develop the necessary competence. As an alternative, the system could allow diagnosis at a lower level--one that would more closely adhere to the domain knowledge of the user. For example, consider the following interaction:

Is gasoline going into the carburetor?

(1) Yes, gas is going into the carburetor
(2) No, gas is not going into the carburetor
(3) I don't know how to check for gas.

When the user selects option 3, it indicates to the system that this particular user is not very knowledgeable about the domain and that either a different type of diagnosis must be performed or tutoring must be provided. If tutoring is offered, it must bring the user to a level at which the questioning can be resumed at the point where it ended, or allow the user to leave the program with the assumption that the tutorial was the desired consultation. If lower level diagnostics are required, then new rules must be generated to provide adequate reasoning and explanation facilities for the user. These rules could be either transparent to the user or used directly to set the value of the parameter about which the user needed further direction.

To demonstrate these three different approaches, decision trees were developed. Figure 2 shows how a tutorial would fit into the present rule structure. As indicated, the diagnostic session would be suspended and an explanation session would be initiated to enlighten the user about the parameter appearing at the present node. In Figure 3, the user enters an alternative diagnostic tree that will communicate at a lower level to obtain the value of the parameter. This branch continues directly into the original branch that would have been next and becomes part of the tracing mechanism used in the explanation. Figure 4 shows the final configuration, which provides a tutorial for the user but exits the program without returning to the node at which the separation from the consultation session occurred.

The first and third examples (Figures 2 and 4, respectively) can be implemented by using shell features or by cleverly coding rules. The second example (Figure 3) is implemented directly with rules. Each of these examples affects the explanation facilities in a different way. In example 1, the explanation facilities would change while the user is within the tutorial portion of the program. In this case, the level of user domain knowledge is below that necessary to understand the explanations; the expert consultation is suspended temporarily while the user is tutored to a level at which the system explanations are clear. The programmer must develop rules and sequences of explanations that will be able to recurse to the user's level and bring about a higher domain knowledge--up to that necessary for an understanding. Within this tutorial, though, the rule structure must be developed carefully so that the automatic explanations provided by the shell system are clear and understandable by the user. Once the user is returned to the level at which expert consultation had been suspended, there will be nothing remaining of the rules in the tutorial that had occurred.

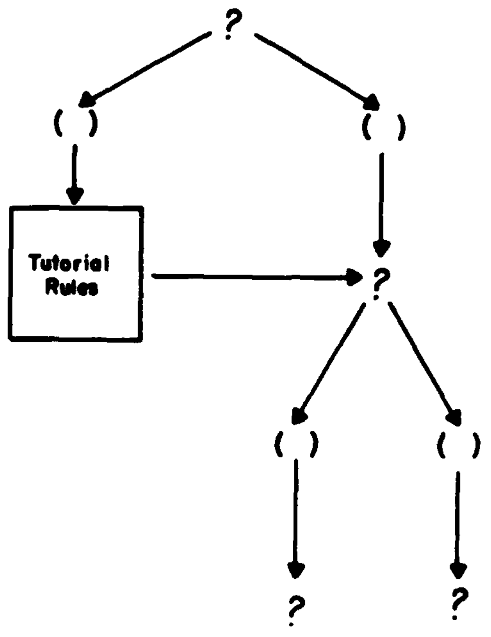

**Figure 2. Model 1.**

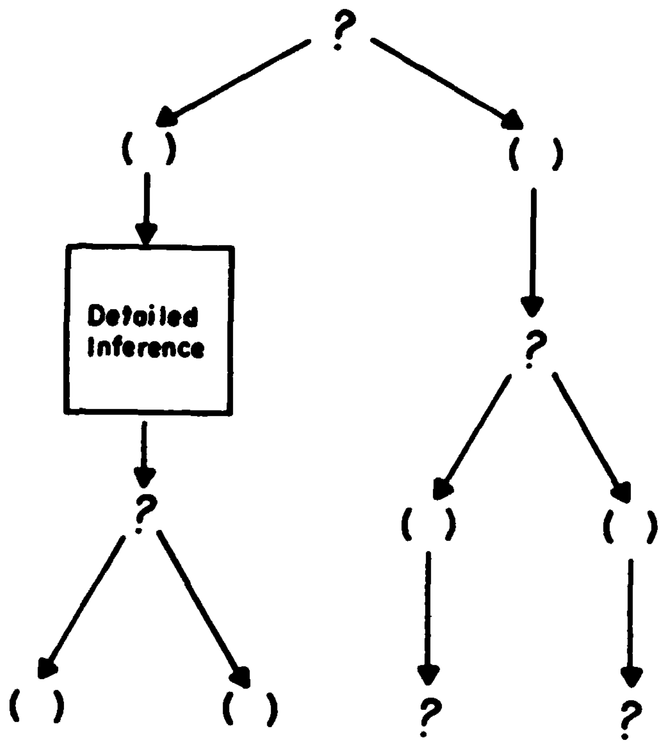

**Figure 3. Model 2.**

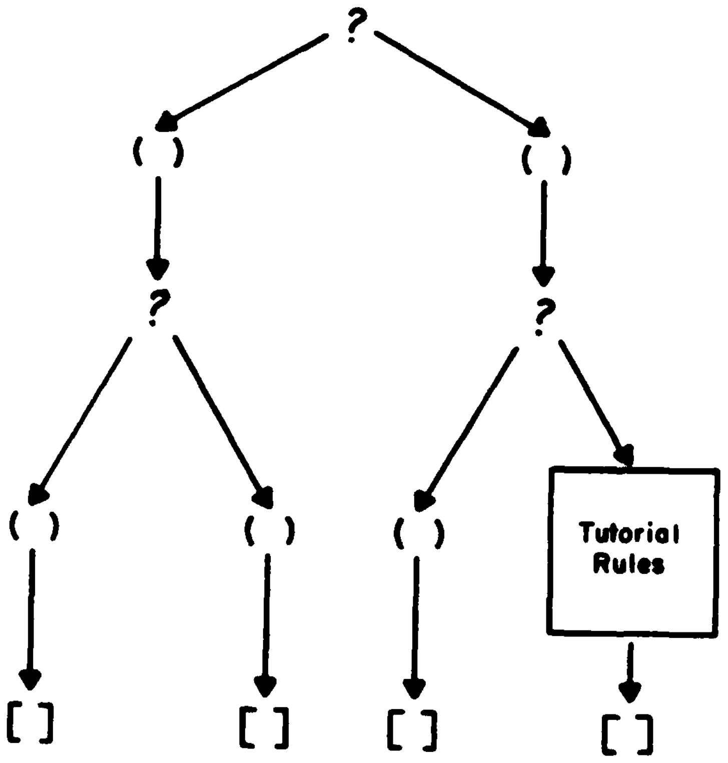

**Figure 4. Model 3.**

In the second example, the user's domain level determines the rules used to pose questions and explanations that are at a lower level of domain knowledge. The explanations generated by the shell will include information at that level throughout the entire consultation and there will be no separation of tutoring and consulting.

In the third method, the value of the parameter would be found similarly to that in the second method, but after this value has been determined by another method, it will be accepted by the program as the conclusion and no further consultation will be provided. These three methods allow the user's domain knowledge level to be detected and the sequence of questions and explanation to be altered. This is done by either bringing the user's knowledge level up to a point at which the explanation facilities are clear or by altering the explanations and questions to conform more closely with the user's domain knowledge. Use of these techniques in shell systems may either offer potentially better alternatives or cause difficulties, depending on the flexibility of a particular shell to implement these models with a suitable user interface.

# 5 MODEL IMPLEMENTATION

To test the models proposed in Chapter 4, they were implemented on an actual expert system developed at USA-CERL using the ESP/Advisor shell. The ESP/Advisor is a commercially available expert system development shell available from Expert Systems International. The application for which the shell system was used is the repair of built-up roofing systems. This expert system was developed through the prototype stage to allow realistic testing of the models. The program is called "Expert System Program for Roof Maintenance (ESROM)."

The ESP/Advisor shell allows the programmer to present textual information about the domain at any time during a consultation session. The shell is based on PROLOG code, and the programmer can add PROLOG to the system to accomplish procedures that are not available in the package. This feature is referred to as "open-ended architecture" and increases the flexibility of the system. The knowledge base is written into a "knowledge representation language" file that is compiled into a knowledge base in a form that the PROLOG inference engine can handle. The syntax of the knowledge representation language allows numerical and string variables of several different types, a feature that increases the flexibility and categories of knowledge that can be represented. These variables include facts, categories, numbers, and phrases.

Facts are binary values that can be associated with a yes or no answer. Numbers can be either integer or floating-point decimals, as well as scientific notation. Categories can take any number of string values specified by the programmer whereas phrases are variable and can assume user-defined values.

ESP/Advisor has a set of logical operators that can be used to define the logical relationships between entities. The following operators are available:

- OR
- XOR
- AND
- NOT.

The symbols and keywords used in the knowledge representation are shown in Figure 5.

An example of rule 1 in Chapter 4 would be:

```
{starter_condition <> normal  and lights <> normal}
'Your battery is either dead or does not have enough'&
'charge.  You should have it checked.'.
```

## Application

The prototype diagnostic expert system for roof repair was used to demonstrate and implement the models for multilevel user domain knowledge. This system is not yet operational in the field, but it could conceivably be developed to that level. The

| | | | |
|---|---|---|---|
| title | section | xor | ensure |
| fact | phrase | number | category |
| range | options | rule | rules |
| askable | using | if | reference |
| and | or | not | true |
| false | quit | do | use |

| | | | | |
|---|---|---|---|---|
| & | .. | @ | ' | : |
| ' | . | = | <> | > |
| < | =< | >= | + | - |
| * | / | ( | ) | # |

**Figure 5. Symbols and keywords in ESP/Advisor.**

application is a fairly typical classification system with constraints involving environmental conditions, materials, and damage levels. Figure 6 shows the classification tree for built-up roof repair.

## Development for Implementation

The decision model shown in Figure 2 was implemented by using the "ensure" command in ESP/Advisor. This feature is used when a mandatory condition is to be imposed before the user can proceed further in the system. In the decision model, the user's domain knowledge is not sufficient so that further consultation is necessary. The user is then sent through a tutorial section of the program and is not allowed to return to the consultation until it is indicated that enough knowledge has been gained to bring the user to an appropriate level. The control for this section of the program begins with the statement:

```
{ leak_type = unsure } reference teach_types.
```

If the user indicates that he/she is not sure what type of leak is present, then this statement will transfer him/her into the tutorial section. In the present version of the system, no online tutorial is given, but reference to a manual is offered for further learning. The system then ensures that the user has achieved a sufficient level by asking for the leak type again. This prompt is performed by the following statement:

```
{ ensure leak_type <> unsure} ''.

{ ensure leak_type <> not_known} ''.
```

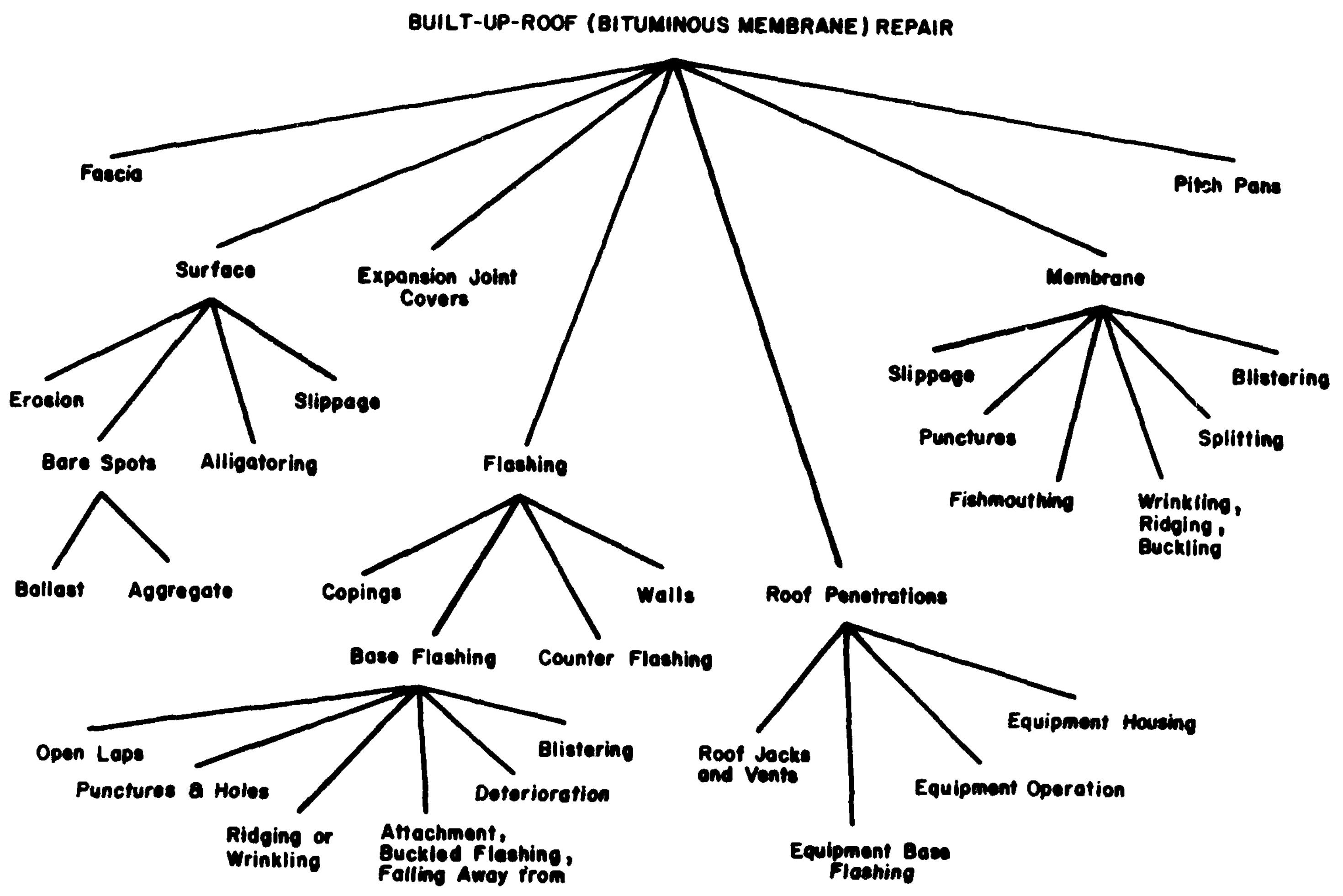

Figure 6. Built-up roofing domain.

This control statement will actually allow the user to change the parameter value of leak_type, reflecting a change in his or her domain knowledge level. The program continues at this point as though the tutorial never occurred (within the context of the rules).

To implement the decision model in Figure 3, production rules were written directly into the code to become part of the rule-tracing facilities. In this model, when the user's domain knowledge is detected to be too low for the present level of questioning, a new set of rules that is less dependent on the user's knowledge enters the decision tree. In addition, the explanation facilities provided with the rules will include knowledge that will be at a lower level and will provide tutoring for the diagnostic process. This model is activated when the user has indicated that the cause of splitting in the built-up roof is unknown. At this point, rules that contain knowledge about the causes of splitting are tapped to ask the user about other, more easily identifiable, parameters which are then employed to directly identify the cause of the splitting. This section of the expert system is controlled by the following statement:

```
{ leak_status <> emergency_repair and split_cause =
    unknown } reference infer_split_cause.
```

In this section of the program, four parameters are identified that will indicate the cause of the split. The explanations and reasoning are available to the user and can provide valuable tutoring to increase the user's domain knowledge about splitting causes.

The implementation of the decision model shown in Figure 4 forced the user into a diagnostic mode that provides tutorial information but, upon finding the original parameter value, ends the consultation session. This model was activated when the user was unable to locate a leak and needed assistance from the expert to find it. The concept was that the system would give the user information with which to go up onto the roof and physically locate the leak. It was evident, therefore, that the user would not be using the system until this action was performed and, as a result, would not be consulting the system again until after returning with the vital information about the location of the leak. This section of the program is controlled in the main body by the statement:

```
{ not location_known } reference find_leak.
```

The section for finding leaks consists of a rule of thumb that can be used to identify a probable location of the leak. By looking at nine parameter values, the system can make a reasonable guess at what is causing the leak. The user can ask the system to explain the facts and reasoning, thus obtaining a good tutorial on the topic of locating leaks. Upon recommending a place to look for the leak, the session is terminated and the user must return later with the leak location in order to proceed.

Each of these three decision models is applicable, given certain user requirements. These involve the physical nature of the particular application and the characteristics of reasoning required throughout the consultation and tutorial process. Although these decision models may provide some additional explanation facilities for the user, they still depend on the features of the shell system used and the possibilities for modeling the user's requirements within the general framework of an expert system.

Finally, the user interface is an important component to develop further.

Consider the example system presented in Chapter 4. After reaching a conclusion by unifying all parameters in rule 5, the system generates the following reasons for that conclusion:

> When the starter is working normally
> and the gas gauge indicates that there is gas
> and gas is going into the carburetor
> and there is no spark at the spark plugs,
> Then it is usually the case that there is a
> malfunction in the ignition system.

Upon analysis, the only parameter that has any relationship with the conclusion is the last one. In fact, a lack of spark alone indicates that there is a malfunction in the ignition system. It is evident that the rule actually has two types of knowledge in it:

1. Knowledge about the causality of the mechanism

2. Knowledge about the diagnostic procedures to identify malfunctions.

Yet, in the development of an expert system, these two types of knowledge are intermingled, causing confusion to the user about the logic of a conclusion or heuristic. Expert system shells should provide a mechanism to incorporate these two types of knowledge so that the user interface can be improved.

# 6 CONCLUSIONS AND RECOMMENDATIONS

This study has reviewed the state of the art in expert systems development for user groups with a wide spectrum of domain knowledge and proposed models for designing such systems. Three decision-tree models were developed and implemented on a prototype expert system for roof repair.

Results show that it is possible to provide a suitable range of knowledge levels for users. This approach will be especially useful in building expert systems for users with unknown domain knowledge levels. However, while these techniques improve current user explanation facilities, they do not address the sources of the problem: the limitations of the available shell systems and a poorly developed understanding about the representation and presentation of knowledge.

Rules used to guide decision trees actually contain two types of knowledge--that about the causality of a mechanism and that about the diagnostic procedures. Both types of knowledge must be incorporated into an expert system to make it truly user-responsive. Without careful attention to including all levels of logic about potential conclusions, there is a risk of confusing some levels of user, thus defeating the system's purpose.

To further improve the explanation facilities, a method of integrating deeper knowledge into an abstract domain theory for providing tutorial information would be of great value. This feature would eliminate the need to cleverly code decision models that have separate limitations, as mentioned above. Such a feature would likely be a network that uses frames and is able to make analogies between the specific application and general scientific principles. This type of system would be complicated and may be beyond the scope of short-term expectations.

For the present, decision models provide an additional tool for expert system developers to improve the explanation facilities and tutorial capabilities of currently available shell systems. They offer a method for improving applications until better shell systems are developed and an understanding of the communication process is substantially increased.

It is recommended that USACE continue studying expert systems technology due to its tremendous potential applicability in military construction. In addition, the prototype built-up roof repair system could be refined for immediate benefit in the field, and the concept could serve as a model for other diagnostic tools.

## REFERENCES

Barr, A., and E. A. Feigenbaum, *The Handbook of Artificial Intelligence*, Vols I through III (HeurisTech Press, Stanford, CA, 1981).

Berry, D. C., and D. E. Broadbent, "Expert Systems and the Man-Machine Interface," *Expert Systems* (October 1986).

Buchanan, B. G., "Expert Systems: Working Systems and the Research Literature," *Expert Systems* (January 1986).

Buchanan, B. G., and R. O. Duda, "Principles of Rule-Based Expert Systems," *Advances in Computers*, Vol 22 (Academic Press, in press).

Harmon, P., and D. King, *Expert Systems* (John Wiley and Sons, 1985).

Hayes-Roth, F., and D. A. Waterman, *Building Expert Systems* (Addison-Wesley, 1983).

Shortcliffe, E. H., *Computer Based Medical Consultations: MYCIN* (North-Holland, Amsterdam, 1976).

## UNCITED REFERENCES

Clancey, W. J., "The Epistemology of a Rule-Based Expert System--A Framework for Explanation," *Artificial Intelligence*, Vol 20, (1983).

Stefik, M., et al., "The Organization of Expert Systems, A Tutorial," *Artificial Intelligence*, Vol 18 (1982).

Tello, E., "Raw Power for Problem Solving," *PC Magazine* (April 16, 1985).

Winston, P. H., and B. K. Horn, *Lisp* (Addison-Wesley, 1984, 1981).

## USA-CERL DISTRIBUTION

Chief of Engineers
ATTN: CEEC-CE
ATTN: CEIM-SL
ATTN: CECC-P
ATTN: CECW
ATTN: CECW-O
ATTN: CECW-P
ATTN: CECW-RR
ATTN: CEEC
ATTN: CEEC-C
ATTN: CEEC-E
ATTN: CERD
ATTN: CERD-C
ATTN: CERD-M
ATTN: CERM
ATTN: DAEN-ZCE
ATTN: DAEN-ZCI
ATTN: DAEN-ZCM
ATTN: DAEN-ZCZ

USAEHSC, ATTN: Library 22060

US Army Europe
AEUES 09081
Allied Command Europe (ACE)
ATTN: ACSGEB/Engr
ATTN: SHIHG/Engr

8th USA, Korea (19)

ROK/US Combined Forces Command 96301
ATTN: EUSA-HHC-CFC/Engr

US Military Academy 10966
ATTN: Dept of Geography & Computer Science

AMC - Dir., Inst., & Svcs.
DEH (23)

DNA ATTN: NADS 20305

FORSCOM
FORSCOM Engineer, ATTN: Spt Det.

NARADCOM, ATTN: DRDNA-F 01760

TARCOM, Fac. Div. 48090

HQ, TRADOC, ATTN: ATEN-DEH 23651

Fort Belvoir, VA 22060
ATTN: Canadian Liaison Officer
ATTN: British Liaison Officer
ATTN: Australian Liaison Officer
ATTN: French Liaison Officer
ATTN: German Liaison Officer
ATTN: Water Res Support Center
ATTN: Engr Studies Center
ATTN: Engr Topographic Lab
ATTN: ATZA-TE-SU
ATTN: STRBE-BLURE

CECRL, ATTN: Library 03755

AFESC, Tyndall AFB, FL 32403

CEWES, ATTN: Library 39180

NAVFAC
ATTN: Facilities Engr Cmd (9)
ATTN: Naval Civil Engr Lab. (2)

NCEL 93043
ATTN: Library (Code L08A)

US Government Printing Office 22304
Receiving Section/Depository (2)

National Bureau of Standards 20899

Defense Technical Info. Ctr. 22314
ATTN: DDA (2)

101
1/89